This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LES.2015.2461626, IEEE Embedded Systems Letters

Framework for a selection of custom instructions for Ht-MPSoC in area-performance aware manner

Bouthaina Dammak*, Mouna Baklouti†, Rachid Benmansour*, Smail Niar*, Mohamed Abid†
*University of Valenciennes and Hainaut Cambrésis, 59300 Valenciennes, France
{bouthaina.dammak, rachid.benmansour, smail.niar}@univ-valenciennes.fr
†National School of Engineers of Sfax, 3042 Sfax, Tunisia
{mouna.baklouti, mohamed.abid}@enis.rnu.tn

***Abstract*—Using application-specific instructions for Heterogeneous MPSoC allows to find a good performance/energy trade-off. For MPSoC architecture executing different multimedia applications, we expect a large number of potential custom instructions. In order to explore the potential of all these instructions, we propose to identify the similar critical computations to be executed on hardware accelerators (HWA) shared between processors. Depending on the running applications in one side and their needs in performance and area usage on the other side, private and shared hardware accelerators are attached to the different cores. This leads to a large architectural space exploration. In this paper we propose an FPGA-based framework capable of identifying the configuration of HWA targeted to an MPSoC architecture. Our framework incorporates a hardware accelerators sharing methodology to optimize area/performance trade-off. The comparison of framework-estimated results and real measurements proves the efficiency of our framework.**

I. Introduction

Heterogeneous Multiprocessor System-on-Chip (Ht-MPSoC) architectures have been emerged in recent years as an important class of very large scale integration (VLSI) systems. An Ht-MPSoC architecture combines a set of embedded processors, several accelerators (audio,video, etc.), memory peripherals, peripherals and interconnection networks. The complexity and heterogeneity of Ht-MPSoC have made them a suitable platform for new multi-media embedded applications. Designing such complex architecture in ASICs was always an efficient solution for optimising system performance. In fact, the maximum complexity of modern ASICs has grown from 5,000 gates to over 100 million. However, the major drawbacks of using ASICs are the typically higher ASIC unit costs and NREs (non-recurring engineering) costs. In addition, ASIC is specific to the application for which it has been designed. This means that modified version of the application will no longer be able to be implemented on the ASIC. The ability to update circuit functionality is achieved by FPGA (Field Programmable Gate Array) technologies. Cyclone V from Altera, Zynq from Xilinx and SmartFusion2 from Micro-Semi are examples of FPGA-based Ht-MPSoC. These architectures include one or more hard-cores and/or soft-cores and up to 500K of reconfigurable Functional Units.

In typical modern Ht-MPSoC, a large number of identical or different applications are simultaneously running on the different processors. For these applications, several computational tasks are candidates to be implemented as Hardware Accelerators (HWA) and invoked by application-specific instructions. The primary problem to integrate the entire number of HWA is the incurred area overhead. As the total FPGA available resources is limited, designer may not be able to exploit the full potential of all HWA for the running applications. For this reason, it is important to use our proposed HWA sharing methodology for Ht-MPSoC architectures. Our methodology consists in finding common computational tasks (patterns) between the concurrent tasks of the applications executed by the various processors. These patterns are then implemented on the FPGA by a reduced number of HWA shared between processors. In the proposed solution, the HWA customization and the HWA sharing degree results in a large design space. In this paper, we propose a framework that identifies common tasks that are candidates to be customized on HWA. Various works proposed resource sharing for custom instructions such as the works presented in [1] and [2]. Our work differentiates itself in many aspects. First, these works treat the selection of fine-grained hardware modules for custom instructions implementation whereas our work focuses on selecting coarse-grained hardware accelerators. Second, unlike the cited, according to the required performance, we consider the implementation of each custom instruction on HWA with or without hardware resource sharing. Third, as we are targeting multimedia applications, where multiprocessors architectures are the most suited platforms in this context, we consider the custom instruction selection for a multiprocessor architecture. In contrast, the mentioned works consider only single processor architecture.

In order to explore the large space of HWA and HWA sharing degrees, our framework integrates a Mixed Integer Linear Programming model (MILP) to identify the HWA configuration of each pattern. This allows designers to come up, in short time, with optimal configuration for an optimised area usage and a fixed performance gain. This is performed via estimating the area usage and performance gain of different possible configurations of the space of solutions in order to find the optimal one. To reduce the time to search the optimal (local optimum) solution, the framework is based on an iterative approach. Such process stops when it is able to find a good solution. Thus, the generated solution is a local optimum MPSoC configuration that satisfies the required performance. In our experiments, we have reinforced the efficiency and the accuracy of the framework by employing a real application.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LES.2015.2461626, IEEE Embedded Systems Letters

II. Proposed Accelerators Sharing Methodology

A. Proposed Sharing Methodology and area saving

Figure 1 shows an example of 3-processors architecture running different applications, each of which contains T1 and T2 as computational tasks (Figure 1.a). While the area resources, named A in figure 1, are limited to 20 units (A=20), only T1 or T2 could be integrated as private HWA for P1, P2 and P3 processors (Figures 1.b). A HWA is in a private configuration if it is coupled to only one processor.

Most of existing embedded applications, such as multimedia, telecommunication or automotive applications, use a same set of critical tasks. Matrix operations, convolutions and filters are frequently used in such applications. For Ht-MPSoC that does not use hardware-sharing, private HWA are implemented to execute the computation of different custom instructions. The proposed sharing approach enables to share HWA of custom instructions performing similar computations. This means that different processors can be coupled to the same HWA. This optimization will avoid bloating the FPGA resources with large number of HWA. We call a pattern the computational task existing on different applications. The pattern identification offers a range of possible sharing optimization.
In Figure1.a, the different applications have same heavy computational tasks (T1 and T2). Based on our proposed approach, a reduced number of HWA for each pattern can be implemented and shared among the processors. The sharing degree defines the number of processors sharing the same HWA. Figure1.c is a possible shared configuration. For T1 pattern, the architecture has one private HWA coupled to P1 and 2-degree shared HWA coupled to P2 and P3. For T2, a 3-degree shared HWA is used and shared between P1, P2 and P3 processors. This configuration consumes 16 area units and provides a reduction of 52% when compared to a private configuration.

B. Impact of Hardware Sharing on Performance

The HWA sharing reduces the area usage but might affect the performance. In fact, as more the sharing degree is increased, the delay to access the shared HWA may increase. Depending on the sharing degree and the processors that share the same HWA, the latency may improve or decline performance improvement. In Figure2.b, sharing T1 between P1, P2 and P3 results a high delay on P2 to execute the shared HWA. This configuration minimizes the area usage but results a higher execution time on P2. In Figure 2.c, the execution time of T1 is enhanced on all processors. In this configuration, the delay to access the shared HWA is negated by adding a private HWA coupled to P1.
As we can see, a fully private configuration bloats the area resources and an aggressive sharing may degrade the performance. Between these two configurations, a very large space of configurations has to be explored.

III. Proposed Framework

In this section we present our proposed framework (Figure3). The aim of our framework is summarized as follows:

- Consideration of different applications. All the processors can execute the same application or different applications.
- For each task, both software and hardware solutions are considered.
- For each HWA, the space of exploration is bounded by a fully private solution for each processor and a fully shared solution between all the processors. According to designer constraints, our framework generates the local-optimal architecture.
- Estimation of area usage and performance gain for the resulted architecture.

A. Applications profiling and Computational Tasks (CT) identification

Our proposed framework starts with compiling and profiling the different applications of the Ht-MPSoC architecture. Application profiling is an important step since it determines the most computational tasks. Embedded system designers are provided with different CAD profiling tools. These profiling tools are classified into three main categories: software-based, hardware-based and FPGA-based tools [3][4][5]. For FPGA-based embedded systems, FPGA-based profiling (FPGA-BP) tools have proved better results compared to the other profiling tools [3] [4]. Thereby for our work we use (FPGA-BP) tools to compile and profile applications.
To select the computational tasks to be candidate for custom instructions implementation and the configuration of their HWA, our framework is based on an iterative approach. We used an iterative approach to save time and efforts by implementing only a sufficient number of computational tasks as HWA, which provide the required performance. In fact, the time needed to design a computational task as HWA might differ depending on the complexity of that task and can reach a couple of weeks. After profiling the different applications running on multiple processors, we have to identify the computational tasks. A task is considered as computational if it consumes more than C% of the overall application execution time. For the first iteration, we compare the highest profiling percentage values of tasks from one application to another. The least value will be considered as the initial value of C. Each new iteration decreases the value of C to the minimum percentage of execution of the next less computational tasks. Thereby, each iteration adds more computational tasks to be explored together with the previous ones until the space exploration generates a feasible solution.

B. Pattern identifications and pattern library

We define a pattern, a computational task existing in different applications. The pattern identification step of figure 3 consists on analysing the identified computational tasks of the previous step to assign the different tasks of similar computations to the same pattern. In each new iteration, the new computational tasks are analysed to identify similarity with the previous defined patterns and/or to add new patterns. The new identified patterns are implemented as HWA in order to determine their information (area usage, performance acceleration when implemented on a HWA). The pattern library is updated in each iteration to include information of new patterns and/or to update information of existing pattern re-identified

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LES.2015.2461626, IEEE Embedded Systems Letters

in new computational tasks . The pattern identification step is currently a manual process, and will be automated in future work.

C. Space exploration

This step finds out the configuration of the local-optimal architecture satisfying the designer constraint. A MILP formulation is proposed in [6] and aims to identify the Ht-MPSoC architecture that minimizes the objective function (Equation 1) and satisfies the performance constraint (Equation2). Our MILP model has as input the information stored in the pattern library and looks for solutions that satisfy the required performance and then it generates the optimal one. So, the optimal solution is the configuration that implements the patterns that provide the best area-performance trade-off.
The objective function is calculated based on the area usage of each pattern if implemented on HWA (a_j) and the area overhead occurred when the pattern is implemented on shared HWA ($a_{overhead}$). For each processor, the acceleration of Equation2 is calculated based on the acceleration of each pattern T_j executed on this processor when implemented on HWA ($tacc_j$) and the delays R_{ji} and $t_{overhead}$ to access the shared HWA of T_j.

$$Total\_Area = \sum_{j=1}^{m}\sum_{i=1}^{n} x_j\left(\frac{a_j}{\sum_{k=1}^{n} y_{jik}} + \alpha_j a_{overhead}\right) \quad (1)$$

$$acc_i = \sum_{j=1}^{m} x_j tacc_j - x_j(R_{ji} + \alpha_j t_{overhead}) \geq limit_i \quad (2)$$

In Equation1, n denotes the number of processors and m the number of patterns. We denote by $N = 1, 2, ..., n$ and $M = 1, 2, ...m$, respectively, the set of processors and the set of patterns. For our MILP (Mmixed Intger Linear Programming) model, x_j is a binary decision variable that denotes whether the pattern T_j is implemented on HWA or not. The y_{jik} decision variable, $j \in M, i \in N, k \in N$, is a binary variable that denotes whether the HWA of pattern T_j is shared between processors P_i and P_k or not. α_j, $j \in M$, is a binary variable that denotes if the pattern T_j is implemented on a shared HWA.
In Equation 2, acc_i, $i \in N$, is the execution time gain for processor P_i and R_{ji}, $j \in M, i \in N$, is the delay of processor P_i to access HWA of T_j. The outputs of cplex resolution are the decision variables that determine the patterns to be executed on software or private and/or shared HWA. Once the optimal configuration is generated, the designer can identify if a pattern would be integrated as custom instruction (x_j variable) and the sharing degree of its HWA (y_{jik} variable). If the model exploration is unable to find a solution, the designer has to decrease the C parameter in order to increase the number of explored patterns. This step is repeated until the model generates a feasible solution.

IV. Experimental Results

In this section, we describe results of applying the proposed framework to an 8-Ht-MPSoC architecture executing jpeg-codec application. We targeted synthesis to a Xilinx Virtex V. Performance measurement and area usage are presented respectively in terms of clock cycles and area units. Xilinx tools provide all GNU/GCC tool chains to compile, link and profile applications for Xilinx supported platforms. The inputs of our framework are the jpeg-encoder and jpeg-decoder applications. We start by compiling and profiling both applications on an 8-microblaze architecture. in which four processors compute the encoder application while the four others execute the decoder application. The output of profiling step is a performance summary of functions that are executed on microblaze processors. For this architecture, the required performance gain has been fixed to 30%. In our framework, the space exploration process searches the custom instructions that satisfy the performance constraint while minimizing the logic area usage. During this search phase, the HW accelerators sharing is considered. The framework was able to generate a solution in the second iteration. For the first iteration the value of C was set to 17% to include the first most computational tasks of the encoder and decoder applications (HDCT and IHDCT tasks). The HDCT and IHDCT functions consist on multiplication of 1*8 matrix with an 8*8 matrix. Thus, we associated one pattern for HDCT and IHDCT tasks (noted HDCT/IHDCT pattern). For this first iteration, cplex was not able to generate a solution that satisfies the required performance. In the second iteration the value of C was decreased to 15% to add the VDCT and IVDCT tasks as computational tasks. Both tasks consist on multiplication of 8*8 matrix with an 8*1 matrix. Thanks to identified similarities, both tasks have been assigned to the same pattern (VDCT/IVDCT pattern).
For each iteration, the identified patterns have been designed in VHDL, implemented using Xilinx ISE (version 12.4), and integrated on a microblaze-based architecture to obtain the library information (area, start-time, end-time, performance gain). For each iteration, cplex returns the result in approximately 4 seconds. In the second iteration, the generated AHt-MPSoC configuration consumes 84 area units and satisfies the required performance. Figure 4 presents the AHt-MPSoC architecture of the generated solution. This architecture consists of 8 microblaze processors and two shared HWA for HDCT/IHDCT pattern and a fully-shared HWA for VDCT/IVDCT pattern. The first HWA of HDCT/IHDCT pattern is shared between five processors (P0,P1, P2, P3, P4) and the second one is shared between P5, P6 and P7. These shared HWA are connected to bus processors through bridges. This architecture is implemented on Virtex V FPGA (XC5VFX70T) and obtained results for area usage and speed up are in accordance to those obtained by our framework. This comparison shows the efficiency and accuracy of our framework .

V. Conclusion

A design space exploration framework for the rapid selection of custom instructions for FPGA-based Ht-MPSoC architecture has been proposed. The framework incorporates a HWA sharing methodology to optimize area-performance trade-off. Comparison of framework estimated results and real measurements on FPGA show the accuracy and efficiency of proposed framework.

VI. Acknowledgment

This work was supported by CMCU project, funded by the Tunisian Ministry of Higher Education and Scientific Research (MESRS) and the French Ministry of Foreign Affairs and International Development

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LES.2015.2461626, IEEE Embedded Systems Letters

REFERENCES

[1] K. Mehdi, Y. Amir, N. Hamid, A. Ali, and P. Massoud, "A new merit function for custom instruction selection under an area budget constraint," *Design Automation for Embedded Systems*, 2013.

[2] Siew-Kei, T. Srikanthan, and C. Clarke, "Selecting profitable custom instructions for areatime-efficient realization on reconfigurable architectures," *IEEE Transactions on Industrial Electronics*, 2009.

[3] J. G. Tong and M. A. S. Khalid, "Profiling cad tools: A proposed classification," in *The 19th International Conference on Microelectronics, December 2007*.

[4] J. Tong and M. Khalid, "Profiling tools for fpga-based embedded systems: Survey and quantitative comparison," *Journal of Computer*, June 2008.

[5] R. Patel and A. Rajawat, "A survey of embedded software profiling methodologies," *International Journal of Embedded Systems and Applications (IJESA)*, Decembre 2011.

[6] B. Damak, R. Benmansour, M. Baklouti, S. Niar, and M. Abid, "Design space exploration for customized asymmetric heterogeneous mpsoc." in *DSD*, 2014, pp. 50–57.

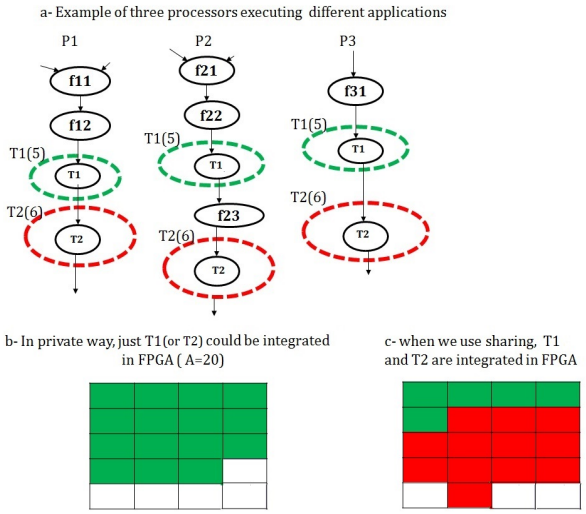

Fig. 1. Illustrative example of benefits of HWA sharing. T1 and T2 are computational tasks executed on P1, P2 and P3. The HWA of Tj(a) consumes a area units in FPGA

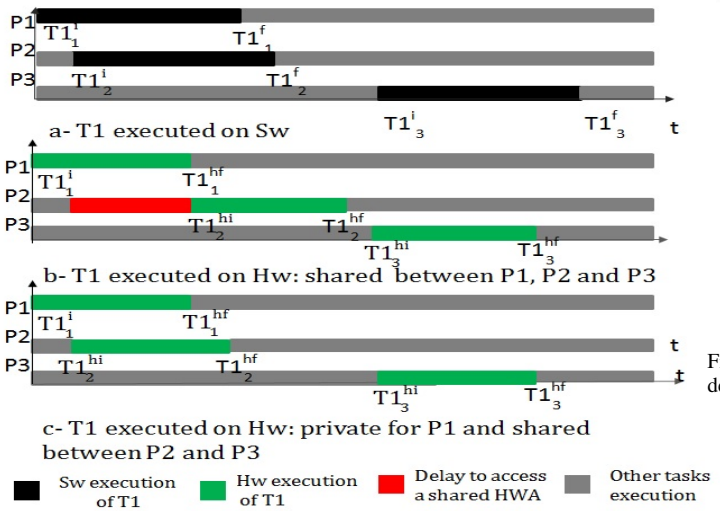

Fig. 2. Different execution configurations for T1 pattern

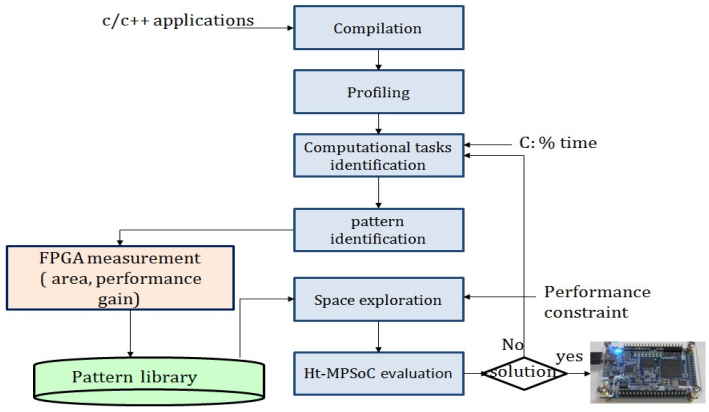

Fig. 3. Proposed Framework

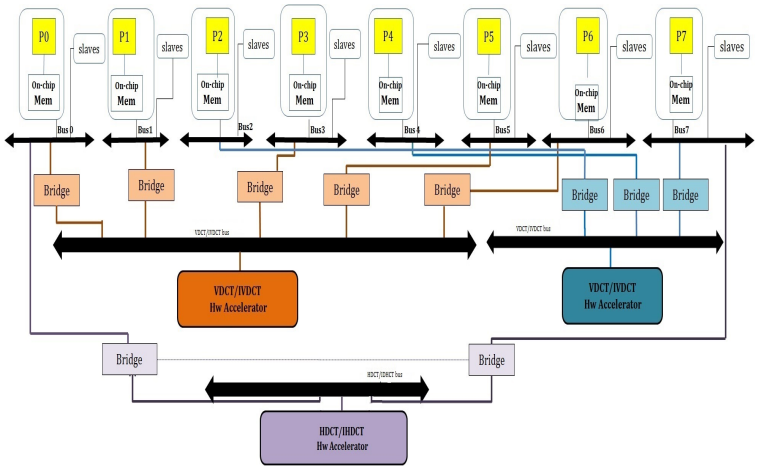

Fig. 4. The framework-based generated architecture for Jpeg encoder and decoder applications.